

DTNServ

A Case for Service Classes in Delay Tolerant Networks

Michael Demmer
CS268 Class Project, Spring 2004
demmer@cs.berkeley.edu

1 Introduction

Networked systems can be characterized by a contract (either explicit or implicit) between an application or other data source and the networking layer. Abstractly, this contract dictates what information applications must provide to the network as part of a data delivery request, and in return, indicates what level of “service” the network will provide to perform the data delivery.

For example, the IP protocol requires an application to specify the source and destination IP address for each packet, and the network provides “best-effort” end-to-end datagram delivery semantics. On the other hand, TCP requires that the application also specify the destination port number and implicitly the packet order, and in return, the network¹ provides in-order, reliable delivery. In either case, when the network cannot perform the desired service, some form of an error indication is returned to the application.

Quality of Service (QoS) systems such as integrated services[3] (intserv) and differentiated services[2, 11] (diffserv) implement a more flexible contract by offering the application the option of varied *Service Classes*. A service class represents an option for the contract between the application and the network that is then multiplexed onto an underlying delivery protocol (in this case IP). The historical intent of these systems is to enable support in the Internet for applications that require more enhanced delivery semantics than best-effort datagram delivery or retransmission based reliable protocols. The canonical example applications for QoS systems are real time multimedia applications that depend on strict delay

and jitter bounds for packet delivery.

Delay Tolerant Networking[5, 7] is an emerging research area that implements a framework for networked services in a variety of “challenged” environments. Some example environments include deep-space systems, characterized by extremely long round trip times and periods of disconnectivity due to orbits, and network infrastructure in developing regions, characterized by inconsistent power and communications infrastructure. In general, the long and unpredictable intermittence of links in these environments renders many traditional networking approaches inadequate.

Unlike the Internet, whose basic building block is the IP datagram service, DTN systems are built around the bundling protocol[12]. Bundles are variable length (application specified) data packets that are forwarded through an overlay routing fabric within the DTN. In contrast to IP routers, DTN routers are intended to implement relatively long term store-and-forward semantics for bundle forwarding to ameliorate the outages of the underlying network links. Similar to IP, the contract between DTN applications and the network requires applications to specify the communication endpoints in each bundle, and the network offers best-effort delivery within the store-and-forward network. Note that there are certain other options that affect delivery semantics (e.g. custody transfer) that are provided by the DTN framework but are largely irrelevant to the remainder of this discussion.

This paper makes a case for the addition of a service class framework to Delay Tolerant Networking and offers an initial design proposal (DTNServ). The remainder of the paper will proceed as follows: Section 2 will outline this case. Section 3 outlines a proposed set of traffic classes to be offered by DTNServ. Section 4 describes the requirements for a reservation

¹For the purposes of this discussion, the transport, networking, and data link layers can be considered “the network” in aggregate.

signaling protocol between DTN router. Section 5 outlines the design goals and relevant work for bundle transmission scheduling and admission control. Section 6 presents some opportunities for future research within the realm of DTN service classes, and finally, Section 7 concludes the discussion.

2 The Case for DTNServ

This section presents the case that for the inclusion of service classes into the DTN architecture. It seems clear that the notion of a quality of service “guarantee” within a DTN is in many ways an oxymoron. By their very nature, DTNs are not reliable, and the goal of DTNServ is not to attempt to circumvent or mask away this unreliability.

Instead, the goal of DTN service classes is fundamentally to enhance the communications interface between the DTN applications and the network layer. To this end, DTNServ provides a mechanism by which the network will guarantee to the application: a) whether the network can provide a desired level of service, b) notification if a previously offered service is no longer available, and c) potentially when the application may expect to receive the service in the future. In turn, the application provides the network with semantic information as to its preference for bundle transmission and as its expected traffic patterns, enabling more efficient forwarding decisions.

The case for DTNServ is the claim that service classes are an effective means to enhance the communications contract between applications and the DTN networking layer, and that this enhanced contract is beneficial to a wide range of applications. This is fundamentally different from traditional Internet QoS systems whose primary goal is the fine grained control over the scheduling of packet transmissions.

2.1 Session Abstraction

The DTN bundling protocol[12] specifies the requirements for the programming interface that applications use within the DTN (see Table 1).

Specifically, the `send` operation requests the transmission of a bundle and returns either an error if the bundle request is invalid or a message token. The token can only be used to request the cancellation of the transmission (assuming that the bundle has not

already been forwarded to another custodian). Note that this is a pure datagram interface, as there is no notion of a “session” between endpoints, and each bundle transmission is treated as an independent entity.

One modification required for the addition of service classes is to augment this interface with the abstraction of a DTN session. As a basic definition, a session is a sequence of bundle transmissions between two application endpoints. Each bundle contains the source and destination endpoint identifiers, the identifying information necessary to classify it into its session. Note that in the DTN terminology, a communications endpoint ID specifies both a node and a particular destination application on that node. This is similar in nature to the combination of an IP address and a TCP or UDP port number; the former identifies the host while the latter demultiplexes to a particular application on that host.

Though the session abstraction is not an advantage in and of itself, as will be shown in the following sections, the notion of sessions within the DTN fabric is a required primitive to establish the service class framework.

2.2 Delivery Information

The primary motivation of the addition of service classes is as a structured mechanism for application feedback as to the state of the network. Most traditional networks implement some form of network feedback; ICMP messages and TCP ACK packets are examples of this type of function in the Internet. These network indications are relayed to Internet applications through an API such as the UNIX `socket` interface. Furthermore, QoS enabled Internet systems also include feedback mechanisms as to the success or failure of a reservation request, and for the (unexpected) cases in which a flow is no longer able to be handled by the network.

A central claim of DTNServ is that DTN applications can benefit from (and some may even be dependant on) similar feedback information. For example, one proposed application for DTNs in developing regions is the periodic broadcast of weather forecast and commodity price information to rural agricultural regions[13]. In this case, the source application may be interested to know if there are any sectors that have not received any price information within a maximum expected delay, say one week. If this oc-

Operation	Description
Send(source endpoint communications ID, destination endpoint communications ID, report endpoint communications ID, class of service, delivery options, lifespan, send token binding, payload)	Initiate a bundle transmission. Returns either a SendError indication or a SendToken that is used for future transactions on this bundle.
Cancel(send token)	Cancel a previously submitted transmission request.
Register(delivery failure action, registration token binding, destination endpoint communications ID)	Register interest for delivery of incoming bundles to a particular destination.
Deregister(registration token)	Cancel a prior registration.
Poll(destination endpoint communications ID)	Explicitly request delivery of any arrived bundles.

Table 1: Bundle application programming interface

curs, an administrator may need to determine where in the network the failure occurred so as to take corrective action.

The current bundle protocol includes one such feedback mechanism, in that an application can register to receive status report bundles whenever its bundles are forwarded by any router within the DTN network. Routers that implement this feature (which is not required) should create a status update bundle at the time of forwarding and send it back over the DTN to the sending application.

While these bundle status updates are sufficient to implement many of the motivating scenarios for DTN service classes, it is a cumbersome and inefficient mechanism. Status reports provide acknowledgement of the expected actions, not the failure scenarios. All the classification and measurement burdens are left to the application to deduce (typically from the lack of a forwarding acknowledgement) that there was a delivery problem in the network. Furthermore, any changes in routing topology must be inferred from the application from a changing pattern of status updates.

As will be shown in more detail below, a core primitive of the DTNServ interface is the mechanism and protocol by which the network informs an application if requested delivery characteristics cannot be met by the current state of the network. This interface allows the application to specify the exact situations in which it wants to get notification of network conditions. Thus rather than requiring the application

to sift through a potentially large set of positive acknowledgements to determine if there was an unexpected failure, the service class system informs the application only of the meaningful situations in the event of an unsatisfied request.

2.3 Traffic Flow Information

As will be shown in more detail in Section 3, the DTN service classes include in their specification a description of the expected network traffic characteristics of the flow itself. The reservation signaling protocol propagates this information to the routers along the path of that flow.

This traffic flow information can then be used by the routers to improve their own operation through capacity planning, bundle scheduling, and link scheduling. Unlike the completely reactionary nature of a best effort router, a router equipped with service class reservations has advance notice of its expected traffic flow. While of course the actual traffic flow may not match the expectation, the expected prior knowledge is still beneficial to the router for provisioning.

As will be explored further in Section 6, DTN routers have a qualitatively different task than traditional routers due to the large timescales of their delivery responsibilities. The store-and-forward capabilities and long periods of disconnectivity lead to an opportunity to spend considerable time exploring delivery options for bundle delivery responsibilities. The per-flow expectation information that comes about through the

use of service classes imparts the router with significant insight as to the future traffic. This may, for example, trigger a router to submit a network “probe” down a potential route.

2.4 Enhanced Services

A final motivation for adding a service class layer to the DTN is to provide a means by which applications that are not intrinsically delay tolerant can still be supported by a DTN deployment.

DTN applications are generally architected to be tolerant of periods of network disconnectivity. In general, this design causes the application developer to make certain design tradeoffs to avoid round trips and blocking inter-node request messages, given that a peer node may be unreachable for significant periods of time. These applications can be classified as *intrinsically delay tolerant*, and can easily be authored to use the bundling protocol.

It is clear, however, that a qualitatively distinct second class of applications are those that are intrinsically not delay tolerant. Obvious examples are IP telephony and online chat, in which a fully connected network path is required between two end hosts.

Abstractly, networks can be modelled as a directed graph, where vertices represent nodes and edges represent links. For a DTN, at a given point in time, only some of the edges are connected. Examining only these connected links, there are a number of subgraphs within the overall contact graph that are “pockets of connectivity”.

A final goal of DTNServ is to support applications that are not intrinsically delay tolerant on these connected subgraphs. As will be shown in the next section, one of the proposed service classes is specifically intended for probing and allocating a fully connected path within the subset of connected nodes. The mechanisms for maintaining session state for service classes can then be leveraged as the means to determine the continued existence of this end-to-end path, thereby enabling the support for applications that are not intrinsically delay tolerant.

3 Service Classes

Historically, QoS systems focus primarily on the distinction between some form of guaranteed service

(e.g. hard and soft real-time in intserv, assured and premium service in diffserv) versus best effort service. Generally, the provisioned flows are constrained to be constant bandwidth and sometimes have delay and burstiness bounds.

As the previous section has outlined, the main goal of DTNServ is not focused on providing guarantees about packet delivery, but rather as a communications mechanism between the network and the application. As such, DTNServ requires a new service class taxonomy. This new set of service classes covers some traditional notions of provisioned traffic flow, yet also expresses the specific characteristics of DTN traffic flow.

3.1 Best Effort Class

The *best effort* class is the basic DTN service class and corresponds to the current contract between the network layer and the application. Specifically, the application has no constraints on the timing or size of bundle transmission, and the network makes no guarantee of ability to deliver those bundles, nor does it offer the application any specific feedback as to the relative success of bundle transmission.

In DTNServ, all bundles that are not specifically classified to one of the other service classes are by default tagged to the best effort service class, requiring no additional interface to the application.

3.2 Allotted Class

The *allotted* class is the first true DTN service class. The specification of flows of this class consists of the parameters outlined in the following table:

source	Source endpoint identifier.
destination	Destination endpoint identifier.
interval	Time interval for bandwidth allocation and report feedback.
allocation	Data rate (in bytes per interval).
min_delivery	The amount (in bytes per interval) of useful data.

Table 2: Allotted service class parameters

The **interval** and **allocation** parameters are the basic elements of the class specification and define the bandwidth (i.e. $\frac{allocation}{interval}$) and the acceptable latency

for packet queuing (i.e. the interval). By accepting an *allotted* flow, the network’s part of the contract guarantees that the application be notified if the network is unable to satisfy the bandwidth request. In return, the application gives the network some information that outlines the acceptability of queuing delays.

The `min_delivery` parameter is an example of a mechanism in which the application provides information to help optimize the network usage. The intent of this parameter is to allow the application to specify a lower bound for useful packet delivery within the given interval. This information can be used by a router in cases where the router does not have capacity to forward the entire allocation, but may be able to send some subset of queued bundles. By consulting the `min_delivery` parameter, the router knows whether to forward the partial allocation or not. An application that depends on an entire transmission being received would then set `min_delivery` equal to `allocation`, and a downstream router then knows not to waste capacity sending part of the allocation since it can know that the ultimate delivery of the partial payload will be of no use to the receiving application.

Note that the application can at any time transmit more data than its allocation. The measure of bandwidth conveyed in the class description can be used as essentially just the key to determine at what level the network reports to the application about its ability to forward traffic, it is not a constraint on the overall traffic of the application. Bundles that arrive out of profile are by default classified as *best effort*.

3.3 Periodic Class

A number of proposed DTN applications involve the periodic transmission or broadcast of information. Again, one example is the broadcast of weather and/or commodity price information from a well-connected city to a number of rural villages. In these applications, typically each message subsumes the previous one, as it is a more current information report. In addition, the period and bandwidth requirements of the traffic flow are generally well known and relatively constant.

These types of applications should use the *periodic* traffic class. The parameters for a flow of this class are outlined in Table 3.

<code>source</code>	Source endpoint identifier.
<code>destination</code>	Destination endpoint identifier.
<code>routing_type</code>	Policy for bundle forwarding (i.e. unicast vs multicast).
<code>period</code>	Time interval between message transmissions.
<code>bundle_size</code>	Size of each bundle.
<code>max_delay</code>	The maximum tolerable delay between transmission and delivery.
<code>squashing</code>	Should bundles squash in network queues.

Table 3: Periodic service class parameters

It can be noted that the parameters of this flow are similar to those of the *allotted* service class, in that both specify a time interval and a total transmission expectation. The main differences lie in the ways in which a router treats bundles of the class when the next hop link is disconnected and the thresholds at which the router determines that the service class is not satisfiable.

Periodic flows can take advantage of a proposed new feature that effects in-network “squashing” of bundles. When a bundle reaches a router where the next hop is disconnected, the standard store-and-forward router will store the bundle awaiting the link re-establishment. If, however, another bundle marked with the “squash” bit arrives for the same flow, the router can discard the previously stored bundle, replacing it with the more recent arrival.

This feature is described in Figure 1, showing a router with two outgoing queues for bundles. The bundles in the first queue have been classified as email messages and therefore are all stored awaiting output. However, the second queue is classified as a flow of weather update information, and therefore the older forecasts are discarded when newer updates arrive. In general, this feature improves the overall network efficiency, since in the intended applications for this feature, older bundles do not add information in the presence of a newer one.²

The second way in which the *periodic* and *allotted*

²Note that the criteria for squashing is not strictly based on timestamps. In the proposed implementation of this feature, each bundle that uses the squashing feature will be augmented with an opaque squashing preference field in the bundle header, assigned by the application. The router will always keep bundles with the higher squash id, allowing the application degrees of freedom to control the squashing of its bundles.

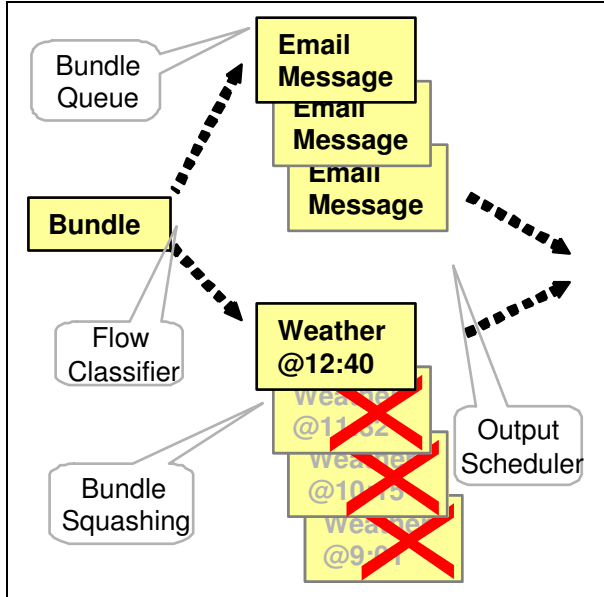


Figure 1: Bundle Squashing Feature

classes differ relates to the triggers used by the network to determine when the class requirements can not be satisfied. The `max_delay` parameter is the mechanism used to convey applications' requirements for transmission latency. If a bundle from the periodic class is not received by the destination after this threshold has been passed, then the network will indicate as such to the application.

3.4 Provisioned Class

As mentioned in Section 2.4, one of the central motivations for adding a service class framework to DTN is the ability to support applications that require connected end-to-end communication. The traffic class used by these applications is the *provisioned* class. The parameters for this class are listed in the following table:

<code>source</code>	Source endpoint identifier.
<code>destination</code>	Destination endpoint identifier.
<code>bandwidth</code>	Rate of transmission of the flow.
<code>delay</code>	Optionally, the delay bound for the flow.
<code>bypass</code>	Indication whether the data will be routed via bundles or not.

Table 4: Provisioned service class parameters

It should be noted that the provisioned class is fundamentally not a delay tolerant traffic class, rather it corresponds with the more traditional QoS real time traffic classes, in that it represents an end-to-end traffic flow of constant bandwidth and optionally tight delay bounds. In particular, packets corresponding to flows of this class are **not** necessarily store-and-forward.

By including the mechanism to allocate a more traditional end-to-end traffic flow, DTNServ is able to leverage the common mechanisms of the signaling protocol, flow classification, network state maintenance, and bandwidth allocation policy. This allows a mix of application types to be run on the same DTN infrastructure deployment.

4 Signaling Protocol

A core requirement of any service class framework is the signaling protocol used for establishment and maintenance of service class state within the network. Given that the key motivation of DTNServ is to enhance the communication contract between the application and the network, this requirement is all the more critical for DTNServ.

This section will outline the requirements for the signaling protocol for DTNServ, describe some existing approaches to QoS signalling, and present some of the open challenges in the protocol's implementation.

4.1 Protocol Requirements

Abstractly, the signaling protocol and application interface should provide the following operations:

Session Request

To make use of service classes, the application must be able to submit a request to the network to establish a new session based on one of the above service classes. Generally, this operation should encapsulate the service class parameters in a bundle and transmit this special bundle along the path to the destination. Each router along the path performs an admission control process to determine if the new session can be supported by the underlying network resources, based on a bandwidth allocation policy (described further in Section 5.2).

Session Acknowledgement

If the admission control process succeeds at all nodes

along the path, the session request bundle will eventually reach the destination node. At this point, an acknowledgement message should be returned to the origin application to indicate successful session allocation.

Note that due to the intermittent nature of the underlying network, there may be a very long latency between the session request and the acknowledgement.

Session Failure

If the admission control procedure fails at an intermediate router due to lack of resources, or a timeout waiting for response, a failure indication message is returned back along the reverse path to the application. Note that all session requests messages should have timeouts to avoid waiting infinitely for an unresponsive peer to send an acknowledgement or failure message.

Session Maintenance

To control the allocation of session state, all routers should adopt a soft-state policy for per-flow data. As such, all allocated flows must be periodically refreshed by the application. The protocol for session requests may be able to be re-used to implement session maintenance.

Session Teardown

Once an application no longer requires a session, it should be able to submit a teardown request to free up the network resources consumed by the session. Note that due to the soft-state nature of the session protocol, this is not strictly necessary for correctness, since all state would time out eventually. However, given that timeout periods are by necessity longer than the potential link outages, it is beneficial to the network to have an explicit request to clean up session state.

Unsatisfiability Notification

As mentioned in the descriptions of the service classes, each class has criteria to be examined by the network to determine if the admitted flow cannot actually be satisfied. This is equivalent to “missing a deadline” in traditional QoS. The signaling protocol must include a mechanism to propagate this information back to the application. Based on the specifics of the service class parameters, this may or may not implicitly revoke the session.

Route Changes

A session is established along a particular routing path, and if the underlying bundle routing topology changes, the established session no longer correlates

with the path that bundles will take. Therefore, the signaling protocol should be able to adapt to changes in the routing topology through some means. One possibility is to “pin” the route for established sessions, i.e. continue to use the original path if possible, while submitting a new session request along the new path, although this may impose unduly constraints on the router operation. Section 6 explores this interaction in more detail.

4.2 Existing Approaches to Signaling

Several signaling protocols have been proposed in the research community, and some are in active use in the Internet today. Perhaps the most well known is the RSVP [14] protocol of integrated services. In addition, the IETF has recently established a *Next Steps in Signaling* (NSIS) working group to address the evolution of signaling protocols in the Internet. The charter of the group is to standardize an IP signaling protocol, using QoS as the first application. The group has outlined a framework for the development of these protocols[4, 6].

The NSIS framework defines a split-layer signaling design, including a base signaling protocol upon which can be various signaling applications, e.g. QoS, can be layered. The NSIS framework is by design modular and flexible to allow several realization scenarios. For instance, the signaling messages may follow the path of the data packets, or may follow an alternative path of routers that presumably could communicate with the routers actually along the data plane. Also, the architecture has provisions for partial deployment, in that not all routers along the path need support the NSIS protocols.

While the NSIS standardization effort is still at its early stages, it is likely that it will be an applicable framework for the DTNServ signaling protocol. Given that the framework and early drafts of protocol specifications are focused on signaling at the IP layer, any specific implementation may not be directly applicable for all DTN scenarios. However, in cases where peering DTN routers communicate via IP the NSIS protocols may be directly usable on a hop-by-hop basis. Also, the general the signaling abstractions and methodologies should be applicable to DTNServ. Therefore, the DTN service class signaling protocol should mirror the NSIS efforts as closely as possible to benefit from that group’s development efforts.

4.3 Challenges

The design of the DTNServ signaling protocol will clearly need to take into account the challenges that arise from the characteristics of DTNs. While not insurmountable, many of these are open research questions that will have to be resolved for a successful DTNServ implementation.

A common challenge in systems and networked systems is in the determination of appropriate timeout values. A basic tussle exists between the desire for short timeouts to increase responsiveness and for long timeouts to avoid potential false errors due to a slow but successful operation. This challenge is all the more relevant to DTNs, given the wide variety of underlying network conditions and their distinct latency characteristics. As a result, it seems clear that timeout values will need to be adjustable throughout the network based on the underlying conditions.

Another related problem is that of cascading latencies. In general, each hop along a path adds the hop's individual latency to the overall latency of the path. Again, given the wide range of hop latencies in DTNs, it may be extremely difficult to calculate an appropriate timeout for the overall path, and it may be the case that due to the additive timeout effects, the particular deployment cannot support the semantic information that is needed by the application.

A potential approach to solving this problem is to augment the signaling protocol with partial bundle acknowledgement that are returned as a request makes its way through the network. This alleviates some of the burden of calculating a single timeout for the entire path, since individual partial acknowledgements could reset timers to an appropriate level for the given next hop link. However, it seems clear that this approach requires further investigation.

Finally, acknowledgement based schemes assume some degree of bidirectional connectivity. While it many DTN links will likely be two-way, some (e.g. satellites) may not, and again due to long latencies, there may be periods of time in which a two-way link is effectively unidirectional. Though the link will eventually return so as to provide a path for acknowledgement packets, for the period of time that the link is disconnected, it is effectively unidirectional. The effect of this network condition on the various signaling protocols is an open issue that needs further exploration.

5 Bundle Scheduling

To be able to offer more sophisticated traffic classes requires that the forwarding logic within a DTN router be modified to be aware of the traffic class characteristics.

This is important for two reasons – the first is that by the very nature of service classes, not all bundles should be treated equally by the forwarding logic. The second, and perhaps more important, is that the router must be aware of the flow characteristics and (recent) history to determine whether or not the network conditions are sufficient to support the set of flows or if a given flow must be revoked.

This section addresses the general topic of scheduling of bundle transmissions within DTNServ. In general, DTNServ will borrow well known techniques from the networking research literature to accomplish this task. This section therefore does not present any ideas unique to DTNServ, but rather discusses the appropriate techniques to use within DTNServ and in some cases, the issues involved with their adoption.

5.1 Classification

One task which a DTNServ router must implement is the ability to correlate bundles with the service class to which they belong. As described in Section 2.1, a “session” refers to a stream of bundles between any two endpoint identifiers. Given that the bundle specification includes the source and destination endpoints in every transmitted bundle, it is a trivial task to correlate a given bundle arrival with a specific service class.

Beyond the per-flow classification, it is also beneficial to support hierarchical classifiers to be able to aggregate bandwidth allocations for more expressive policies. Several well explored schemes for packet classification exist in the research literature[9, 8]. The flow classification within DTNs is sufficiently similar that it is expected that these schemes can be easily adopted to the DTN system.

In addition to classification of flows, the router must also keep some statistics as to the historical activity of each session for bandwidth management. For the *allotted* class, the router tracks the total amount of bandwidth consumed by a session's bundles within the current interval. This is used to schedule of future

bundles as well as to determine whether or not the router can meet the service request. Similarly, for *periodic* class sessions, the router must keep track of the number of “slots” in which it did not transmit a packet that it expected to, in addition to state as to whether or not it has transmitted the bundle in the current slot.

5.2 Bandwidth Allocation

A basic task that a network router faces is that of allocating the outgoing bandwidth among the set of queued packets. While best effort semantics can be implemented without any knowledge of the capacity of links, to implement an admission control policy and/or rate based flow classification, a router needs to know the capacity of its outgoing links to determine whether or not the link can support the traffic required by the set of admitted flows.

For any allocation to occur, the router must obtain information as to the underlying capacity of outgoing links. This mechanism is largely orthogonal to the problem of DTNServ, and as such, approaches including administrative configuration, link estimation, or more complex signaling schemes will be sufficient.

In some scenarios, a mixture of configuration and estimation is appropriate. Assume a DTN router with a fixed outgoing link, but the power supply at the other router is inconsistent. Thus, when the link the other side has power, the bandwidth capacity of the link is well known and configurable. However, the intermittence of the link is not known ahead of time, and must be estimated to get an accurate estimation of the overall link capacity.

The second aspect of the allocation scheme is the actual distribution of the outgoing bandwidth to the set of queued bundles. Typically, the allocation process involves an administratively configured policy that enumerates the bandwidth proportions to be assigned to various service classes or aggregations of flows. For example, a simple policy would allocate 50% of bandwidth to periodic and allotted flows, reserving the remaining 50% for best effort bundles. The process of allocating the bandwidth is then straightforward based on the policy.

Note that in cases where the link’s schedule of disconnectivity is known, these algorithms should take into account the schedule. When the link is not scheduled, but has unexpected periods of disconnection,

this task is more challenging. This case is explored further in Section 6.3.

5.3 Deadline Assignment

Once bandwidth has been allocated to a session, the router must then assign a delivery order to the session’s bundles to effect the calculated allocations.

Hierarchical packet fair queuing[1] and related work are well-examined scheduling disciplines for service class based routers. The basic algorithms assign deadlines to each arriving packet based a function of the session’s bandwidth allocation. Packets are then served from the queue in earliest deadline first order.

These standard deadline-based approaches should be easily adaptable to the case of DTNServ.

5.4 Calculating Satisfiability

The basic admission control process is a straightforward test between the capacity of the underlying network and the requirements of the session requests. However, this does not fully cover the problem of satisfiability, since a DTN router must be able to handle the case in which the underlying network does not perform up to expectations. This is clearly evident in any sort of intermittent (i.e. not scheduled) contact, as the state of whether or not the link is available is not known in advance to the router.

Thus unlike many traditional QoS systems, DTNServ must have mechanisms to deal with sessions when bundles miss their deadlines. For instance, as noted in the *allotted* service class, the class parameters include the minimum useful amount of data specifically to give information to the router as to how to handle cases when it does not have sufficient capacity for the whole session, but may be able to send part of the session’s traffic.

6 Research Opportunities

This section briefly introduces some of the future research opportunities enabled by the addition of DTN service classes.

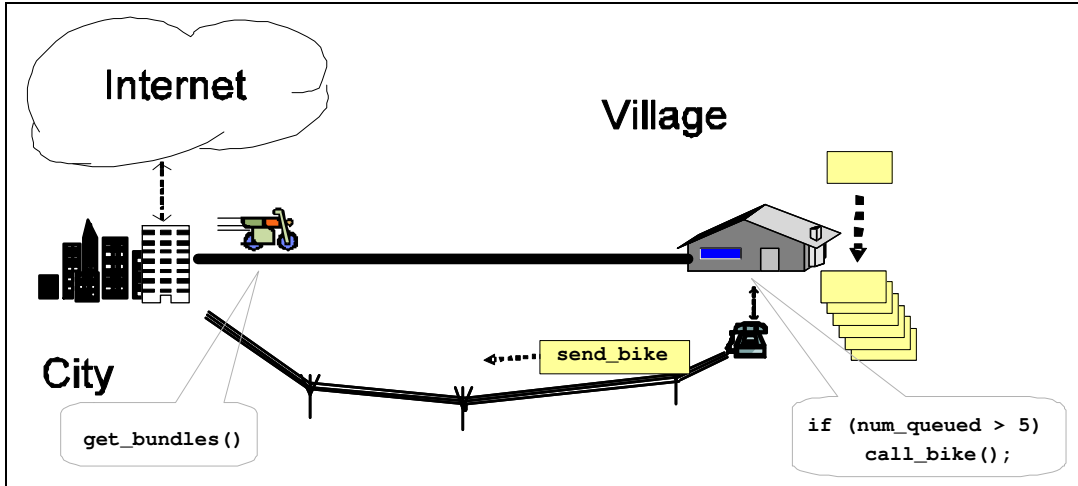


Figure 2: Cross Layer Integration

6.1 Cross Layer Integration

Traditionally, networking systems are characterized by layers as a means of controlling the interface between various code modules and algorithms. This model applies to traditional QoS systems, with respect to the interactions between the QoS systems and the network routing functions. The routing layer generally operates independently from the QoS system, performing routing functions to maintain the best paths within the network. The QoS layer must adapt to route changes and update reservations and path allocations accordingly.

In the case of DTNServ, this layering may not be the best architecture, as there are situations in which a more synergistic relationship between the routing layer and the QoS system is advantageous. The question of what is the “best route” within a DTN depends largely on the metrics in question, given that different links may exist with different costs[10].

An avenue for future research in the area of DTN service classes lies in the potential for cross-layer interaction between the service class system and the routing layer. The long time scales of DTN service class deadlines give the router a considerable amount of flexibility in choosing a next hop router for a class of packets. For example, a router may issue a small network probe down a set of paths to determine the reservation capability of the paths and then select the best one in response to a session request.

In addition, the service class mechanism can even be

used to modify the network topology itself. Consider the hypothetical example depicted in Figure 2. The DTN router in the village has two outgoing links, a low-bandwidth high cost dialup model and the opportunity of a motorcycle based digital courier with a high bandwidth capacity, but long delay. Conceptually, the deployment would like to only send the motorcycle out to the village if it is worth the delay, i.e. the queued data exceeds some threshold. The DTNServ framework can help provide the router with the necessary insight as to the deadline requirements and predicted future transmissions of the various DTN sessions so as to determine if it should call for the bike to come and collect the outgoing bundle data.

6.2 Multipath Forwarding

Given the unpredictability of DTN contact paths, in some situations, it may be preferred to fragment a bundle and forward it to multiple downstream contacts for eventual reconstruction by the receiver. The application of erasure codes may be particularly beneficial in this situation to allow the receiver to reconstruct the original bundle from a fraction of the total number of fragments.

An open question is how to provide the benefits of service classes within an underlying routing fabric that implements multipath fragment forwarding. The issues involved are similar to those in traditional IP QoS systems that deal with multipath forwarding for

load balancing reasons. Yet traditional approaches such as route pinning are less effective, as they would nullify the advantages of multipath forwarding.

However, as outlined in the previous section, the service class management system and other components of the DTN framework like the routing component may be relatively tightly integrated. This enables the potential for more effective solutions to the multipath problem. For example, a router could divide the bandwidth requirement for a given service class session among the outgoing links proportionally based on the multipath forwarding rules. It could then perform an aggregation process from the individual session feedback responses for the outgoing links to determine an overall response for the given session.

6.3 Intermittence Aware Algorithms

The final area of research for service classes relates to characterizing intermittent links. In traditional networking, a point to point link can be characterized by a (*bandwidth, latency, jitter*) tuple. It is therefore a relatively straightforward operation to map a set of flow requirements onto this overall capacity to determine if the link can satisfy the requirements of all the flows.

A DTN link introduces a new parameter, the *intermittence schedule* that is qualitatively different from these elements, and represents the periods of disconnectivity of the link. It is possible to fold the intermittence factor into the standard parameters, either by increasing the jitter value to take into account the potential outages, and/or by adjusting the bandwidth value to amortize the periods of connectivity and disconnectivity. However, these approaches lose the qualitative information about the link that it truly has two states, and that those states should be treated as distinct.

This unique link characterization parameter requires new design of traffic flow based algorithms. Essentially, the intermittence is a metric as to the rate of change of the network topology, which is traditionally assumed to be fairly static.

For example, Section 5.2 notes the dependence of the bandwidth allocation on estimates (or schedules) of link connectivity. The simple allocation scheme largely ignores the effects of disconnectivity, reacting to an unexpected outage by reporting that the service class is unsatisfiable. However, with more sophisti-

cal estimation techniques that involve an awareness of the intermittence of a link, more informative reports may be possible, such as the future likelihood of a session being supportable by the network.

7 Conclusions

This paper presents the case for service classes within Delay Tolerant Networks, outlines an initial design framework for such a system, and describes some open research challenges and opportunities within this arena. In general, the case is based around the need for a richer contract interface between applications and the DTN fabric, and the design of DTNServ is targeted to the effective communication of network state to and from applications.

References

- [1] Jon C. R. Bennett and Hui Zhang. Hierarchical packet fair queueing algorithms. *IEEE / ACM Transactions on Networking*, 5(5):675–689, 1997.
- [2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475: An architecture for differentiated services, December 1998.
- [3] R. Braden, D. Clark, and S. Shenker. RFC 1633: Integrated services in the Internet architecture: an overview, 1994.
- [4] M. Brunner, R. Hancock, E. Hepworth, C. Kappler, and H. Tschofenig. RFC 3726: Requirements for signaling protocols, April 2004.
- [5] V. G. Cerf, S. C. Burleigh, A. J. Hooke, L. Torgerson, R. C. Durst, K. L. Scott, K. Fall, and H. S. Weiss. Delay-tolerant network architecture. Internet Draft, March 2003. <ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-irtf-dtnrg-arch-02.txt>.
- [6] R. Hancock (editor), I. Freytsis, G. Karagiannis, J. Loughney, and S. Van den Bosch. Next steps in signaling: Framework. Internet Draft, October 2003.
- [7] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of*

the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pages 27–34. ACM Press, 2003.

- [8] Anja Feldmann and S. Muthukrishnan. Trade-offs for packet classification. In *INFOCOM (3)*, pages 1193–1202, 2000.
- [9] Pankaj Gupta and Nick McKeown. Packet classification on multiple fields. In *SIGCOMM*, pages 147–160, 1999.
- [10] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *SIGCOMM*, September 2004. To Appear.
- [11] K. Nichols, V. Jacobson, and L. Zhang. A twobit differentiated services architecture for the internet, December 1997. draft-nichols-diff-svc-arch-00.txt, Internet draft.
- [12] K. Scott and S. Burleigh. Bundle protocol specification. Internet Draft (Proposed), April 2003. <ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-irtf-dtnrg-bundle-spec-03.txt>.
- [13] Technology and Infrastructure for Emerging Regions (TIER). <http://tier.cs.berkeley.edu/>.
- [14] Lixia Zhang, Stephen Deering, and Deborah Estrin. RSVP: A new resource ReSerVation protocol. *IEEE network*, 7(5):8–?, September 1993.